TP1

BDSI 2024/2025

1- Préparer et inspecter les données

Étapes :

1. **Importation des données** :
   - Charger un jeu de données avec `pandas` (`pd.read_csv()`, `pd.read_excel()`, etc.).
   - Exemple de jeu de données : un fichier CSV contenant des informations sur les ventes, les clients, etc.
2. **Exploration initiale** :
   - Afficher les premières lignes du jeu de données avec `df.head()`.
   - Obtenir un résumé statistique des données numériques avec `df.describe()`.
   - Vérifier la structure et les types de données avec `df.info()`.
   - Identifier les valeurs manquantes avec `df.isnull().sum()`.
3. **Vérification de la cohérence des types de données** :
   - Vérifier si les colonnes ont les bons types (`int`, `float`, `datetime`, etc.) par print(df.dtypes)
   - Convertir les colonnes en types appropriés avec `pd.to_datetime()`, `astype()`.

2- Gérer les valeurs manquantes

Étapes :

1. **Identifier les valeurs manquantes** :
   - Vérifier les valeurs manquantes avec `df.isnull().sum()` et `df.isna().sum()`.
2. **Supprimer les lignes/colonnes avec des valeurs manquantes** :
   - Utiliser `df.dropna()` pour supprimer les lignes ou `df.dropna(axis=1)` pour les colonnes.
3. **Imputer les valeurs manquantes** :
   - Imputer les valeurs manquantes avec une valeur spécifique (par exemple, la moyenne) : `df.fillna(df.mean())`.
   - Imputation conditionnelle (par exemple, imputer par la médiane pour certaines colonnes).
   - Utiliser des techniques d'imputation plus avancées comme `KNNImputer` de `sklearn`.
4. **Vérification après imputation** :
   - Vérifier que les valeurs manquantes ont été correctement traitées avec `df.isnull().sum()`.

3- Détection et gestion des doublons

Étapes :

1. **Détecter les doublons** :
   - Utiliser `df.duplicated()` pour identifier les doublons.
   - Afficher le nombre de doublons : `df.duplicated().sum()`.
2. **Supprimer les doublons** :
   - Utiliser `df.drop_duplicates()` pour supprimer les doublons.
3. **Gérer les doublons partiels** :
   - Supprimer les doublons en fonction de certaines colonnes uniquement : `df.drop_duplicates(subset=['colonne1', 'colonne2'])`.

4- Normaliser et transformer les données

Étapes :

1. **Transformation des colonnes numériques** :
   - Appliquer des transformations comme `log()` ou `sqrt()` pour des colonnes avec des distributions asymétriques.
   - Utiliser `df['colonne'] = df['colonne'].apply(lambda x: ...)` pour des transformations personnalisées.
2. **Normalisation des données** :
   - Normaliser une colonne pour qu'elle ait une moyenne de 0 et un écart-type de 1 avec `StandardScaler` de `sklearn`.
   - Utiliser `MinMaxScaler` pour amener les valeurs d'une colonne dans une plage de 0 à 1.
3. **Gestion des valeurs aberrantes** :
   - Détecter les valeurs aberrantes avec des méthodes comme l'IQR (interquartile range) ou les boxplots (`sns.boxplot()`).
   - Supprimer ou transformer les valeurs aberrantes.

5- Traitement des variables catégorielles

Étapes :

1. **Identifier les variables catégorielles** :
   - Identifier les colonnes avec des données non numériques : `df.select_dtypes(include=['object'])`.
2. **Encoder les variables catégorielles** :
   - Utiliser `pd.get_dummies()` pour effectuer un encodage one-hot des variables.
   - Utiliser `LabelEncoder` ou `OrdinalEncoder` de `sklearn` pour transformer les catégories en valeurs numériques.
3. **Vérification de l'impact de l'encodage** :
   - Vérifier que les colonnes ont bien été encodées et qu'il n'y a pas d'erreurs de transformation.

6- Nettoyage des dates et des heures

Étapes :

1. **Conversion des dates** :
   - Convertir une colonne de dates en type datetime avec `pd.to_datetime()`.
   - Vérifier les erreurs dans les formats de dates avec `df['colonne_date'].dtype`.
2. **Extraction d'informations temporelles** :
   - Extraire l'année, le mois, le jour, le jour de la semaine avec `df['colonne_date'].dt.year`, `df['colonne_date'].dt.month`, etc.
3. **Gestion des valeurs temporelles manquantes** :
   - Imputer les dates manquantes ou les combler avec des valeurs par défaut.

7- Visualisation pour détection des anomalies

Étapes :

1. **Visualisation des distributions** :
   - Utiliser `sns.histplot()`, `sns.boxplot()` ou `plt.hist()` pour visualiser la distribution des variables continues et repérer les anomalies.
2. **Visualisation des corrélations** :
   - Utiliser `sns.heatmap()` pour afficher la matrice de corrélation et détecter les variables fortement corrélées.
3. **Tracé des tendances temporelles** :
   - Visualiser l'évolution des variables au cours du temps avec `sns.lineplot()` ou `plt.plot()` pour détecter les anomalies temporelles.